SOFTWARE ENGINEERING LABORATORY SERIES

SEL-81-014

(NASA-TM-84766) AUTOMATED COLLECTION OF SOFTWARE ENGINEERING DATA IN THE SOFTWARE ENGINEERING LABORATORY (SEL) (NASA) 73 p CSCL 09B

N82-29043

Unclas
G3/61 28398

AUTOMATED COLLECTION OF SOFTWARE ENGINEERING DATA IN THE SOFTWARE ENGINEERING LABORATORY (SEL)

SEPTEMBER 1981

NASA
National Aeronautics and Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

SOFTWARE ENGINEERING LABORATORY SERIES SEL-81-014

AUTOMATED COLLECTION OF SOFTWARE ENGINEERING DATA IN THE SOFTWARE ENGINEERING LABORATORY (SEL)

SEPTEMBER 1981

NASA

National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

FOREWORD

The Software Engineering Laboratory (SEL) is an organization sponsored by the National Aeronautics and Space Administration, Goddard Space Flight Center (NASA/GSFC) and created for the purpose of investigating the effectiveness of software engineering technologies when applied to the development of applications software. The SEL was created in 1977 and has three primary organizational members:

NASA/GSFC (Systems Development and Analysis Branch)
The University of Maryland (Computer Sciences Department)
Computer Sciences Corporation (Flight Systems Operation)

The goals of the SEL are (1) to understand the software development process in the GSFC environment; (2) to measure the effect of various methodologies, tools, and models on this process; and (3) to identify and then to apply successful development practices. The activities, findings, and recommendations of the SEL are recorded in the Software Engineering Laboratory Series, a continuing series of reports that includes this document. A version of this document was also issued as Computer Sciences Corporation document CSC/TM-81/6222.

The primary contributor to this document is

Arthur Green (Computer Sciences Corporation)

Other contributors include

William Decker (Computer Sciences Corporation)
Frank McGarry (Goddard Space Flight Center)

Single copies of this document can be obtained by writing to

Frank E. McGarry
Code 582.1
NASA/GSFC
Greenbelt, Maryland 20771

ABSTRACT

This document examines the collection of software engineering data in the Goddard Space Flight Center (GSFC) Software Engineering Laboratory (SEL). The current manual collection of data via software engineering forms is evaluated with regard to what can and cannot be automated. Top level functional requirements for an automated system for the collection of software development statistics are presented.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

Table

SECTION 1 - INTRODUCTION

Software engineering (SE) is a discipline that seeks to provide a more scientific approach to computer software design and development. In order to learn how to develop software more scientifically in the Goddard Space Flight Center (GSFC) Mission Support Computing and Analysis Division (Code 580) environment, the Software Engineering Laboratory (SEL) was created to measure and evaluate the effects of various methodologies in current use (Reference 1).

The stated goals of the SEL can be broken down into the following three major categories:

1. Monitor current project progress
2. Collect SE data to determine how software is being developed
3. Evaluate the effects of various methodologies across several GSFC Code 580 projects, with regard to their impact on software development

One of these major functions is the collection and analysis of SE data. During the last 5 years, the SEL has attempted to collect SE data pertinent to the design and development of several major software systems. The goal of this study has been to determine areas where time and effort has been unproductive and where improved methodologies might be employed to produce a better product.

The data collection instrument consists primarily of a set of six software engineering forms which are filled out on a regular basis by programmers and systems designers involved in a given development project. The forms are supplemented by computer accounting information, code analyzers, personal interviews, and subjective management data.

To date, the data collection and analysis have proven to be costly, time consuming, and subject to inaccuracies. This is primarily due to the manual collection and preparation of the data for entry into a data base management system (DBAM) which performs report generation but very little analysis.

The manual data collection process is a slow and tedious process in which many people (including managers, programmers, analysts, and support personnel) must complete forms, validate the data, and enter SE data into the data base. There is no feedback mechanism for analyzing the data and folding the results back into the projects. Also, human factors, such as programmer motivation (or lack of it), play an important part in the accuracy of the data collected.

Because of these drawbacks to manual data collection, automatic extraction of SE data in the SEL would be very desirable. Even though validation of the collected data would be required, the time currently spent filling out the forms and entering the data would be saved, since the data would be collected and stored on the same machine that the development effort is using. There would be virtually no influence from human factors on the data collected in an automatic mode.

The purpose of this document is to analyze this possibility. Section 2 gives an overview of the current SEL data collection process. Section 3 describes the SEL data that could be automatically collected, and Section 4 discusses the types of SEL data that could not be extracted automatically. Some top-level functional requirements for an online automated data collection system are given in Section 5, and Section 6 presents the conclusions and recommendations resulting from this study.

SECTION 2 - OVERVIEW OF THE SEL DATA COLLECTION PROCESS

This section gives an overview of the data collection process followed in the SEL. Included in the overview is a brief description of the software engineering forms used and the relationship of data collection to the software development process. Also given is a brief discussion of some special considerations in automating the SEL data collection.

2.1 SEL FORMS

The data collection system which has evolved in the SEL consists of a set of six reporting forms which are completed at various stages of software development. These forms are shown in Appendix A and are summarized below.

- General Project Summary--This form defines the scope of the software development problem.
- Component Summary--This form describes the structure of each component (e.g., module or routine) of the software system under study.
- Resource Summary--This form provides manpower charges and computer usage statistics.
- Component Status Report--This form details the activitites of the programmer/designer on each component of the software system.
- Run Analysis--This form provides the results of a given program execution.
- Change Report--This form gives the reason for and a description of each change to the software system.

As mentioned in Section 1, these forms are filled out on a regular basis by the programmers and systems designers involved in a given development project. (See Section 2.1 of Reference 2 for details of the SEL data collection and the software engineering forms.)

2.2 SEL DATA COLLECTION AND THE SOFTWARE DEVELOPMENT PROCESS

The SEL data collection procedure attempts to measure the total resources of the software development process as it exists in the SEL environment. (See Figure 2-1 for an illustration of a typical SEL software development life cycle.) In order for the data collection procedure to be effective, it must monitor development activities throughout the entire software life cycle and not just during design and implementation.

The software development process is divided into a number of serial and distinct functions linked by informal, loosely coupled communication channels between the requirements, design, coding, testing, integration, operation, and maintenance phases. Most of the focus to date has been on monitoring the requirements, coding, and testing phases, with very little effort directed to monitoring the design and maintenance phases.

The existing component phases need to be connected in a more systematic manner. In this way, each area of the development process can be classified according to the type and amount of resources it requires. If an accurate profile of development activities is to be obtained, items such as the programmer's/designer's use of core, central processing unit (CPU) time, and input/output (I/O) activity must be logged during the activity. The types and number of interrupts initiated by the user and their frequency give some indication of development activities in an interactive environment, but they are inadequate when batch procedures are evoked.

2.3 SPECIAL CONSIDERATIONS IN AUTOMATING SE DATA COLLECTION

The degree of automation of data collection is dependent on the following: (1) the sources of data (real and potential)

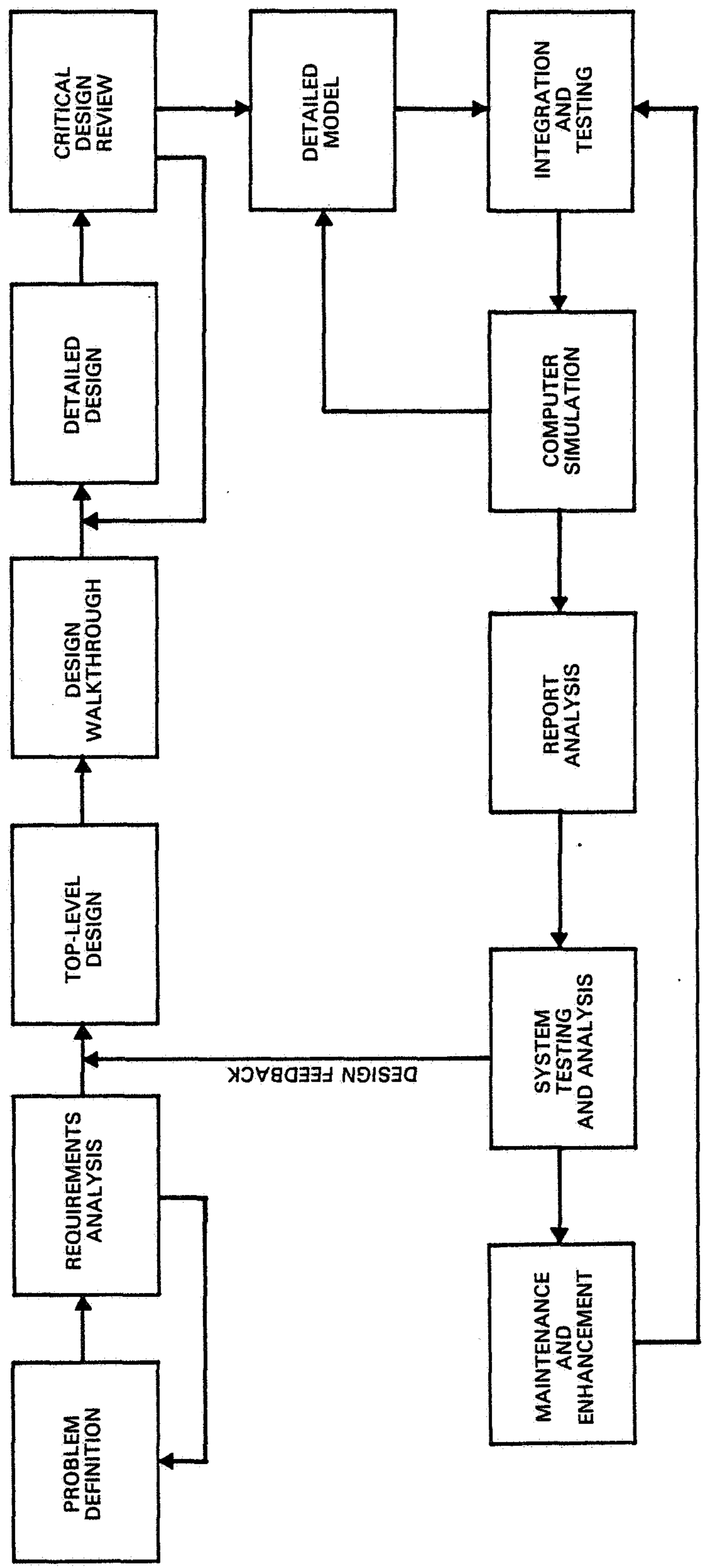

Figure 2-1. Typical SEL Software Development Life Cycle

and (2) the level of system support to be given to the designers and developers of an automated system. Ideally, the data collection should be done at the highest system level possible, rather than as some invoked procedure or called application system. This ensures the uniform application of data collection for all users.

Another special consideration in automating SEL data collection is the case of subjective data. Because software development is primarily a human activity, certain types of subjective information are desirable. However, it is necessary to decouple the subjective data from the automated collection process or, where possible, to restate the goals so that they can be specified objectively. (Subjective data are discussed further in Section 4.)

SECTION 3 - SEL DATA SOURCES FOR AUTOMATIC EXTRACTION

One of the goals of this document is to define the type of SE data that can be collected automatically in the SEL. This section discusses those types of data.

The computers available to SEL users are the Digital Equipment Corporation (DEC) PDP-11/70 and VAX-11/780. These computers are rich in sources of data in their own right. In addition, several software tools and utilities already exist in the SEL which provide other sources of SE data. Table 3-1 gives a lengthy list of current and potential sources of online SE data in the SEL. The remainder of this section summarizes the currently available sources, in some cases providing examples and brief descriptions.

The types of data which could be collected automatically are broken down into the following categories:

- Accounting information
- Keyboard monitor
- VAX object module analyzer
- Requirements analysis tools (MEDL-R, PSL/PSA)
- Programmer workbench
- Text editors
- Program Design Language (PDL)
- Utilities
- Compiler and linker statistics
- FORTRAN Static Source Code Analyzer (SAP)

3.1 ACCOUNTING INFORMATION

Accounting routines generally provide information about resource utilization (such as CPU and I/O usage, direct-access volume usage, and page faults) because their primary purpose is to provide a basis for billing projects. However, most systems allow for user-written accounting routines which collect data for later analysis.

Table 3-1. Sources of Online Software Engineering Data

1. Compiler/asembler statistics (number and type of coding error)
2. Linker/task builder
3. Online debugging tools (ODT)
4. Accounting files
5. Software engineering tools (e.g., PSL/PSA, MEDL-R, CSMR, FINREP, MARS)
6. System error log
7. Overlay descriptor files (i.e., who calls whom)
8. Automated Program Design Languages (e.g., Caine, Faber, and Gordon)
9. Text editors (e.g., ODC)
10. Keyboard monitors (examine each keyboard entry for software engineering information)
11. Programmer workbench
12. Performance measurement and monitoring (e.g., Boole and Babbage)
13. Login/logout information
14. System management records
15. System and user-developed utilities (e.g., PIP, COPY, DIFF)
16. Financial tapes
17. User directory information (good source of change information)
18. Source analyzers (e.g., SAP)
19. Resource estimators (e.g., Price S, Doty, SLIM, GRC)
20. System services (SYS$GETJPI, GETTSK)
21. Error trapping mechanisms (exit handlers)
22. Complexity functions (e.g., Halstead measures)
23. Maintenance procedures
24. Data bases
25. Configuration management systems (CAT)
26. Formal test procedures
27. Dump/trace facilities
28. Cross reference programs

Since the interface with the system already exists on both the SEL PDP-11/70 and VAX-11/780 computers, this area provides one of the most reliable and easily implemented methods of obtaining resource utilization information on a project-by-project basis. Data set information is already recorded whenever a file is opened, scratched, renamed, closed, or processed by end of volume. A SEL enriched accounting procedure could form the basis around which a more comprehensive and elaborate data collection scheme might be built.

The types of information currently available in the VAX-11/780 accounting file are shown in Figures 3-1 and 3-2. Similar types of information are available on the PDP-11/70.

3.2 KEYBOARD MONITOR

Both the VAX-11/780 and the PDP-11/70 provide collections of routines which can be linked with user programs to provide the capability of processing command lines dynamically. The system facilities include, for example, the following:

Routine Name	Description	Function
GCML	Get command line	Retrieves keyboard input
CSI	Command string interpolator	Takes command lines from the GCML input buffer and parses them

This set of software can be used to develop keyboard monitors that examine each line entered at a terminal for SE-related data. When it exists, the SE data would be extracted and stored for later processing and analysis. Because of the high volume of data obtained in this manner, rigorous screening and filtering techniques might be required to extract pertinent SE data. It is, however, an area that warrants further investigation.

1. Message type
2. Message length
3. Final exit status
4. Process identification (PID)
5. Job identification
6. Termination time
7. Account name string
8. User name string
9. CPU time in 10 ms units
10. Total page faults
11. Peak paging file usage
12. Peak working set size
13. Count of buffered I/O operations
14. Count of direct I/O operations
15. Count of volumes mounted
16. Login time
17. PID of subprocess owner
18. Termination message length
19. Job name (batch)
20. Queue name
21. Symbiont page count
22. Symbiont QIO count
23. Symbiont GET count
24. Time job was queued
25. Name of print job
26. Name of print queue
27. Length of print accounting record
28. User message area
29. Job termination
30. Batch job termination
31. Interactive job information
32. Login failure process termination
33. Print job accounting
34. Inserted message
35. Insert message into accounting file
36. Create a new account file
37. Enable accounting
38. Disable accounting
39. Enable selection accounting
40. Disable selection accounting

Figure 3-1. VAX Accounting File and Termination Message Contents

USERNAME	TYPE	START-TIME	ELAPSED-SECS	CPU-SECS	PG-FAULTS	BUF I/O	DIR I/O	VOL-MOUNTS	JOB-NAME	PG-COUNT	GIOS
DEB03	P	25-JUN-1981 10:50:03							TABLE	2	116
AD301	P	25-JUN-1981 10:50:04							SEEK	2	85
ADG01	B	25-JUN-1981 10:52:34	12	4.60		105	74	0	SEEK		
ADG01	P	25-JUN-1981 10:52:46							SEEK	2	85
ADG01	I	25-JUN-1981 10:17:26	2850	176.43	7450	3928	3146	0			
AD301	B	25-JUN-1981 11:01:56	290	248.33		98	1208	0	ZER		
ADG01	P	25-JUN-1981 11:06:46							ZER	2	88
RED02	I	25-JUN-1981 10:39:35	1727	20.76	1070	1835	1166	0			
LND01	I	25-JUN-1981 09:59:07	4485	44.76	2958	3707	2388	0			
RED05	I	25-JUN-1981 11:23:21	119	6.07	613	296	55	0			
LND01	B	25-JUN-1981 11:17:06	591	238.20		2216	7622	0	LIST1		
LND01	P	25-JUN-1981 11:26:57							LIST1	2	79
DEB03	P	25-JUN-1981 11:30:26							TABLE	2	116
DEB03	P	25-JUN-1981 11:37:46							TABLE	2	116
DEB03	I	25-JUN-1981 10:18:13	4895	217.33	16180	3643	2941	0			
GAIL	I	25-JUN-1981 11:08:49	2743	35.77	3634	2907	558	0			
DEB03	P	25-JUN-1981 11:55:11							TABLE	2	63
LND01	I	25-JUN-1981 10:27:29	5611	142.19	7581	4100	1189	0			
RED05	I	25-JUN-1981 12:08:21	296	86.27	8811	1054	635	0			
RED05	P	25-JUN-1981 12:13:13							DEBUG	56	2868
FIY02	I	25-JUN-1981 11:19:23	4512	251.70	9297	6928	6818	0			
LND01	P	25-JUN-1981 12:57:37							INTMN1	54	2763
LND01	I	25-JUN-1981 10:41:54	8379	42.79	2917	8413	584	0			
DEB03	P	25-JUN-1981 13:18:31							DEATTML	6	352
DEB03	P	25-JUN-1981 13:18:37							TABLE	3	149
DEB07	P	25-JUN-1981 13:24:08							FILLSTRE	3	136
DEB03	P	25-JUN-1981 13:24:17							FILLLINE	3	160
DEB03	P	25-JUN-1981 13:24:27							GETWORD	4	151
GMA07	I	25-JUN-1981 13:39:44	270	7.43	881	383	157	0			
RED02	I	25-JUN-1981 14:07:37	317	6.46	144	1191	14	0			
GSS05	I	25-JUN-1981 13:09:35	4164	14.39	500	1881	181	0			
RED05	I	25-JUN-1981 14:14:00	321	1.69	143	177	14	0			
RED03	P	25-JUN-1981 14:20:31							DEBUG	34	2231
RED05	P	25-JUN-1981 14:25:12							DEBUG	5	259
LND01	I	25-JUN-1981 13:08:26	5086	58.36	643	3081	5293	0			
RED03	P	25-JUN-1981 14:33:38							DEBUG	31	2028
GSS05	P	25-JUN-1981 14:36:21							GINIT	12	448
GMA07	I	25-JUN-1981 14:41:13	261	4.10	605	189	96	0			
RED02	P	25-JUN-1981 14:47:37							DEBUG	26	1178
RED02	I	25-JUN-1981 14:13:36	2183	59.93	5060	5290	1010	0			
RED02	P	25-JUN-1981 14:49:40							EVNTS	10	420
RED05	P	25-JUN-1981 14:50:53							LOCKUP	2	60
RED05	P	25-JUN-1981 14:58:04							LOCKUP	3	70
FIY02	P	25-JUN-1981 15:04:22							EXGLOB2	6	355
GMA07	I	25-JUN-1981 14:55:33	823	6.97	928	356	161	0			
GMA07	P	25-JUN-1981 15:09:14							MIKEPROC	2	67
RED05	P	25-JUN-1981 15:14:07							YANG1	4	64
RED05	P	25-JUN-1981 15:16:30							YANG1	2	62
MAD03	I	25-JUN-1981 15:10:56	353	3.47	455	269	42	0			
GMA07	I	25-JUN-1981 15:16:59	151	1.03	106	80	10	0			

Figure 3-2. VAX Accounting File Information

3.3 VAX OBJECT MODULE ANALYZER

The VAX object module analyzer (ANALYZE) provides a description of the contents of an object file or the symbolic information appended to a shareable image file. In describing the records, ANALYZE also identifies errors if they exist. This information is less amenable to further analysis, because its content is sketchier than that given by source code analysis. It is given here as an additional source of SE data.

Figure 3-3 presents an example of the output from the ANALYZE option.

3.4 REQUIREMENTS ANALYSIS TOOLS (MEDL-R, PSL/PSA)

Requirements analysis encompasses all aspects of software development prior to actual system design. The SEL has conducted some ground-breaking studies in this area by examining currently available requirements packages such as the Problem Statement Language/Problem Statement Analyzer (PSL/PSA, Reference 3) and the Multi-Level Expression Design Language - Requirements (MEDL-R, Reference 4). Computer-aided tools such as these can be modified and enhanced to extract relational and hierarchical data from their associated data bases.

The basic concepts in automated requirements analysis are well documented (see References 5, 6, and 7). Requirements analysis seeks to ensure correctness of the end product, unambiguity, consistency, and completeness. If a completely automated data collection system is to be developed, more work must be done to refine and/or develop more tools in this area.

```
*****************************************************************************
**************** Native Object Module Analyzer Version 4.03 ************
***************************  Analysis of file  ***************************
**************** _DBB1:[FDYN.FDY03.ALLOC]TSTSQ1.OBJ;99 ****************
*************************  3-SEP-81 20:02:16 *************************
*****************************************************************************


>>>>>>>>>> RECORD      1 IS A MODULE HEADER    54 BYTES LONG <<<<<<<<<<<
        STRUCTURE LEVEL =  0
        MAXIMUM RECORD LENGTH = 1024
        MODULE NAME IS 'TSTSQ1$MAIN'
        MODULE IDENT IS '01'
        CREATION DATE/TIME WAS 18-Aug-1981 15:34
        LAST PATCH DATE/TIME WAS 18-Aug-1981 15:34

>>>>>>>>>> RECORD      2 IS A LANGUAGE PROCESSOR SUB-HDR    24 BYTES LONG
        ASCII DATA IS:
             VAX-11 FORTRAN V2.2-40

>>>>>>>>>>>>>> RECORD   3 IS TRACEBACK   41 BYTES LONG <<<<<<<<<<<<<<<
        COMMAND     1 IS STORE IMMEDIATE,  22 (DEC) BYTES       STACK=  0
        IMMEDIATE BYTE STREAM (IN HEX) FOLLOWS:

         0   1   2   3   4   5   6   7   8   9
      ----------------------------------------
  0     12  BC  00  01  00  00  0C  0B  54  53
 10     54  53  51  31  24  4D  41  49  4E  12
 20     BE  00


        COMMAND     2 IS STPBB    (  4)                           STACK=  4
        P - SECTION NUMBER =  0
        VALUE STACKED =            0 (DEC)           0 (OCTAL)
                                               0 (HEXADECIMAL)
        COMMAND     3 IS STOPIDR ( 27)                            STACK=  0
        COMMAND     4 IS STORE IMMEDIATE,  12 (DEC) BYTES         STACK=  0
        IMMEDIATE BYTE STREAM (IN HEX) FOLLOWS:

         0   1   2   3   4   5   6   7   8   9
      ----------------------------------------
  0     0B  54  53  54  53  51  31  24  4D  41
 10     49  4E
```

Figure 3-3. Output From the VAX Object Module Analyzer (ANALYZE)

3.5 PROGRAMMER WORKBENCH

The programmer workbench (PWB) concept is generally regarded as a highly specialized computing facility dedicated to satisfying the needs of software developers. In principle, it is a front end which provides a convenient work environment and a uniform set of programming tools across machine boundaries. PWBs have been configured for many diverse hardware environments and have supported development for many target computers.

Recently, GSFC Code 580 has embarked upon the development of phase 1 of a PWB tailored specifically for the Code 580 software development environment (Reference 8). It is similar to the well-known Bell Telephone Laboratories PWB/UNIX (Reference 9). However, because of the continuing need to collect statistics which accurately describe the SEL environment, the development of Code 580 PWB phase 2 provides an excellent opportunity to integrate automated development with automated data collection. The tools and methods used in conjunction with the Code 580 PWB should place high emphasis on SE data collection.

3.6 TEXT EDITORS

Text editors are available in several forms in the SEL VAX/PDP environments. Editors are one of the primary means by which data are created and modified in the development of software. If detailed creation and change information is to be collected, one viable option is to provide text editors that have been modified to extract SE data. Modules which provide summaries of changes made to a given module could easily be coupled with the Code 580 PWB to extract data from interactive sessions and record it for later processing or inclusion in the SEL SE data base.

Some work has already been performed in this area at GSFC. An Online Data Collector (ODC) has been developed, which is, in fact, an SE-related editor (Reference 10).

3.7 PROGRAM DESIGN LANGUAGES (PDLS)

Software development is still largely a manual process. There has been relatively little effort devoted to design validation and analysis. Top-down, structured design has contributed to the formulation which must precede design automation, i.e., it must be known just what constitutes design. Although some initial work has been done by Freeman (Reference 11), there is still little organized knowledge of what a software designer does.

Flow charts and baseline diagrams still remain as the principle method for representing software designs. The machine processable design representation of the Caine, Faber, and Gordon Program Design Language (PDL) system is one of the few automated design tools on the market (see Reference 12).

Once more of the design information is in machine-readable form, more can be done to develop procedures for automatically extracting SE data for the design process. However, it is still not clear how much can be done to formalize software design. This is an important area which needs to be investigated more thoroughly before significant progress can be made towards automated collection of software design statistics.

3.8 UTILITIES

The SEL defines a utility as any component that is generated for the purpose of staisfying some general support function required by other applications software. This class of software contains programs that do not fit into any other category in the software development life cycle.

The SEL PDP-11/70 and VAX-11/780 both support forms of the Peripheral Interchange Program (PIP), which is the primary data manipulation software in the SEL. Utilities such as PIP usually provide statistical summaries on the results of the operations performed or could easily be modified to do so.

Other SEL utilities, such as the VAX Difference Analyzer (DIFF), the DISKUSE utility, and the locally developed FORTRAN cross-reference program (XREF), are examples of the type of support software that already exist in the SEL and that could be incorporated into an automated statistics extraction and reporting system. In the VAX environment, the DIFF utility compares the contents of two disk files and creates a listing (or file) of the records that do not match. A sample execution of the DIFF utility is shown in Figure 3-4. The DISKUSE utility provides data on storage requirements, sorted by project and group. Sample output from this utility is given in Figure 3-5.

3.9 LINKER/TASK BUILDER STATISTICS

The VAX-11/780 linker and the PDP-11/70 both provide data on the structure and content of executable images and shared global areas. The MAP option, when specified, generates data on the following:

- Module name
- Object modules which comprise the image
- Image sections
- Symbols
- Module address
- Module lengths (size)
- Line statistics
- Module creation date
- Language translator that created the module
- Global sections referenced

```
$ DIFF
$_File 1:     [FDYN.SRC]HNDLER.FOR
$_File 2:     [FDYN.FDY03]HNDLER.FOR
********************************************************************
************************* FILE COMPARE UTILITY *********************
************************** DIF -- VERSION 1.12 *********************
********************************************************************


********************************************************************
********************** MERGED LIST OF DIFFERENCES ******************
********************************************************************


FILE SY:[FDYN.SRC]HNDLER.FOR;33
   48          CHARACTER TNAME*(*),PRNAME*12
   49          CHARACTER INPUT*80,OUTPUT*80,TERMI*5,TERMO*5
*****************************
FILE SY:[FDYN.FDY03]HNDLER.FOR;479
   48          CHARACTER TNAME*(*),PRNAME*12,MBX*12
   49          CHARACTER INPUT*80,OUTPUT*80,TERMI*5,TERMO*5
*******************************************************
*******************************************************
FILE SY:[FDYN.SRC]HNDLER.FOR;33
   53          INTEGER*2 ITMHAF(2),ILEN,JLEN,JFLAG
   54    C
*****************************
FILE SY:[FDYN.FDY03]HNDLER.FOR;479
   53          INTEGER*2 ITMHAF(2),ILEN,JLEN,JFLAG,MBXUNT,ICHAN
   54    C
*******************************************************
*******************************************************
FILE SY:[FDYN.SRC]HNDLER.FOR;33
   82    C
   83    C     OPEN MAILBOX UNIT
   84    C
   85          MAILBX = 3
   86          OPEN(UNIT=MAILBX,TYPE='NEW',NAME='MAILBOX.DAT',
   87         * RECORDSIZE=1024,FORM='UNFORMATTED')
   88    C
   89    C     LOAD MAILBOX BUFFER
   90    C
   91          BUFFER(1) = LOC
   92          BUFFER(2) = IFLAG
   93          BUFFER(3) = NARG
   94          NAMCX = NMLNAM
   95    C
   96          DO 10 I=1,12
   97       10 AUTFLE(I) = AUTFIL(I)
   98    C
   99          IF(NARG.LE.0) GO TO 30
  100    C
  101    C     LOAD GLOBAL NAMES IN MAILBOX BUFFER
  102    C
  103          DO 20 I=1,NARG
  104          BUFF(I) = BLANK
  105       20 CALL XTRACT(%VAL(DARRAY(I)),BUFF(I),KLEN)
  106    C
  107       30 IF(IPASS.GT.1) GO TO 50
  108    C
*****************************
FILE SY:[FDYN.FDY03]HNDLER.FOR;479
   82          WRITE(6,123) KFLAG,KERROR,NUMARG
   83      123 FORMAT(' HNDLER:  JFLAG,KERROR,NUMARG = ',3I10)
   84    C
   85          IF(IPASS.GT.1)  GO TO 5
   86    C
*******************************************************
*******************************************************
```

Figure 3-4. Output From the DIFF Utility

```
[FDYN.FDV05.NEST].....................              612
[FDYN.FDV05.NOTE].....................               51
[FDYN.FDV05.PLOT].....................              516
[FDYN.FDV05.PM].......................               42
[FDYN.FDV05.PROCES]...................               24
[FDYN.FDV05.PRPLOT]...................              141
[FDYN.FDV05.QT].......................              363
[FDYN.FDV05.QUOTE]....................               54
[FDYN.FDV05.QUOTAL]...................             1221
[FDYN.FDV05.RADMAS]...................             1713
[FDYN.FDV05.RT11].....................               15
[FDYN.FDV05.SEARCH]...................               51
[FDYN.FDV05.SETPORT]..................               18
[FDYN.FDV05.STAN].....................              258
[FDYN.FDV05.TAPECOPY].................               97
[FDYN.FDV05.TAPC].....................              330
[FDYN.FDV05.TEST2]....................              255
[FDYN.FDV05.TESTGRAPH]................              642
[FDYN.FDV05.TIME].....................               81
[FDYN.FDV05.TIMER]....................                9
[FDYN.FDV05.VT100]....................              579
[FDYN.FDV05.WATT].....................               48
[FDYN.FDV05.XEROX]....................               36
[FDYN.FDV05.XREF].....................              351
[FDYN.FP2]............................             1515
[FDYN.HOLD]...........................             4656
[FDYN.ORBIT]..........................             1221
[FDYN.RADMAS].........................              435
[FDYN.SPC]............................             1979
[FDYN.TEST]...........................              252
                          TOTAL   FDYN            62016

[GMAS]................................             2573
[GMAS.ACCU]...........................               27
[GMAS.CARDTAPE].......................              123
[GMAS.DATCOM].........................               57
[GMAS.MIXED]..........................              222
[GMAS.PROGM]..........................            14469
[GMAS.RADMAS].........................             9367
[GMAS.TEMP]...........................              198
[GMAS.VIUTI]..........................              939
                          TOTAL   GMAS            27975

[GPES]................................             2154
[GPES.SOR]............................              465
[GPES.SOR.STRING].....................               90
[GPES.SOR.FMT]........................              132
                          TOTAL   GPES             2841

[MASC]................................             3606
[MASC.ALAN]...........................               45
[MASC.PAM]............................              786
[MASC.PROREM].........................              654
[MASC.STATREM]........................              390
[MASC.SUMSHARE].......................              207
                          TOTAL   MASC             5688

[SYSMGR]..............................                9
                          TOTAL   SYSMGR              9

                          TOTAL BLOCKS           199417
```

Figure 3-5. Output From the DISKUSE Utility

- Number of virtual pages required
- Base and ending addresses of program sections (PSECT)
- PSECT attibutes
- Library access
- Symbol cross reference
- COMMON block usage
- Stack size
- Image type
- Storage requirements for image
- Number of modules
- Number of global symbols
- Virtual memory allocated
- Overlay descriptor

Sample link output is provided in Figure 3-6.

3.10 COMPILER STATISTICS

The FORTRAN compiler options provide many items of data pertinent to the data collection process. The Storage Map section summarizes information about memory allocation, and the Program Section Summary describes module structure. The Entry Point Summary lists all entry points and their addresses and identifies the section function.

The compiler listing can be used to obtain the following data:

- Program sections
- Entry points
- Variables
- Statement function
- Arrays
- Labels
- Functions and subroutines called
- Total memory allocated
- Module names

```
_DBB1:[FDYN.FDY03.PARM]PARTST.EXE;9                        9-SEP-1981 18:55         LINKER V28.44
                                                    +-----------------+
                                                    ! IMAGE SYNOPSIS !
                                                    +-----------------+

VIRTUAL MEMORY ALLOCATED:                   00000200 000279FF 00027800 (161792. BYTES, 316. PAGES)
STACK SIZE:                                       20. PAGES
IMAGE HEADER VIRTUAL BLOCK LIMITS:                 1.          1. (     1. BLOCK)
IMAGE BINARY VIRTUAL BLOCK LIMITS:                 2.         74. (    73. BLOCKS)
IMAGE NAME AND IDENTIFICATION:              PARTST 01
NUMBER OF FILES:                                  17.
NUMBER OF MODULES:                                70.
NUMBER OF PROGRAM SECTIONS:                       30.
NUMBER OF GLOBAL SYMBOLS:                       1013.
NUMBER OF IMAGE SECTIONS:                         18.
USER TRANSFER ADDRESS:                      00009C00
DEBUGGER TRANSFER ADDRESS:                  80000168
IMAGE TYPE:                                 EXECUTABLE.
MAP FORMAT:                                 DEFAULT IN FILE "_DBB1:[FDYN.FDY03.PARM]PARTST.MAP;1"
ESTIMATED MAP LENGTH:                       117. BLOCKS
                                                 +----------------------+
                                                 ! LINK RUN STATISTICS !
                                                 +----------------------+

PERFORMANCE INDICATORS                          PAGE FAULTS   CPU TIME       ELAPSED TIME
----------------------                          -----------   --------       ------------
    COMMAND PROCESSING:                                 24    00:00:00.33    00:00:01.85
    PASS 1:                                            773    00:00:03.10    00:00:07.44
    ALLOCATION/RELOCATION:                              43    00:00:00.10    00:00:00.52
    PASS 2:                                            314    00:00:01.95    00:00:05.75
    MAP DATA AFTER OBJECT MODULE SYNOPSIS:             151    00:00:02.01    00:00:02.11
    SYMBOL TABLE OUTPUT:                                10    00:00:00.01    00:00:00.17
TOTAL RUN VALUES:                                     1315    00:00:07.50    00:00:17.84

USING A WORKING SET LIMITED TO 300 PAGES AND 140 PAGES OF DATA STORAGE (EXCLUDING IMAGE)

TOTAL NUMBER OBJECT RECORDS READ (BOTH PASSES):   1455
    OF WHICH 620 WERE IN LIBRARIES AND 136 WERE DEBUG DATA RECORDS CONTAINING 4255 BYTES
3911 BYTES OF DEBUG DATA WERE WRITTEN,STARTING AT VBN 75 WITH 8 BLOCKS ALLOCATED

NUMBER OF MODULES EXTRACTED EXPLICITLY            = 2
    WITH 53 EXTRACTED TO RESOLVE UNDEFINED SYMBOLS

45 LIBRARY SEARCHES WERE FOR SYMBOLS NOT IN THE LIBRARY SEARCHED

A TOTAL OF 0 GLOBAL SYMBOL TABLE RECORDS WAS WRITTEN

/MAP/EXEC=PARTST PARTST,GETADD,ALLOC,CKNAMF,[FDYN.HOLD]RADMAS/OPTIONS
```

Figure 3-6. Sample Link Statistics

- Program section attributes
- Module size
- Compile time

Sample compiler data is shown in Figure 3-7.

3.11 DIRECTORY INFORMATION

Files maintained on the PDP-11/70 and VAX-11/780 are referenced through directories. The directory for each user contains the following information:

- File protection
- Size in blocks
- Owner
- Date and time created
- Date and time last revised
- Expiration date
- File attributes
- Record format
- Record attributes
- File organization
- Total of in-use/allocated blocks
- Number of files
- Version numbers

Additionally, Digital Command Language (DCL) commands and system utilities such as SRD can be used to obtain sorted, specialized subsets of data for a given user identification code (UIC). A sample directory listing with the full option is shown in Figure 3-8. The system file analyzer (SFA) can also be used to display formatted dumps of disk files, as shown in Figure 3-9.

```
PARTST$MAIN                                       8-SEP-1981 18:55:36    VAX-11 FORTRAN V2.2-40                   PAGE  6
                                                  8-SEP-1981 16:54:23    _DRB1:[FDYN.FDY03.XYPLOT]PARTST.FOR;59

LABELS

    ADDRESS   LABEL        ADDRESS   LABEL        ADDRESS   LABEL        ADDRESS   LABEL        ADDRESS   LABEL        ADDRESS   LABEL

      **      5              **      6              **      10             **      15         1-00000011  444'       1-00000000  555'
  1-00000021  1111'          **      10001

FUNCTIONS AND SUBROUTINES REFERENCED

ALLOC          DISPLA         EFRONT         GETADD         S[illegible]T

TOTAL SPACE ALLOCATED = 18857 BYTES

COMMAND QUALIFIERS

  FORTRAN /LIST PARTST

  /CHECK=(NOBOUNDS,OVERFLOW)
  /DEBUG=(NOSYMBOLS,TRACEBACK)
  /F77  /NOG_FLOATING  /I4  /OPTIMIZE  /WARNINGS  /NOD_LINES  /NOMACHINE_CODE  /CONTINUATIONS=19

COMPILATION STATISTICS

  RUN TIME:           4.11 SECONDS
  ELAPSED TIME:       4.76 SECONDS
  PAGE FAULTS:        457
  DYNAMIC MEMORY:     78 PAGES
```

Figure 3-7. Sample Compiler Data

```
DIRECTORY _DBB1:[FDYN.FDY03.ALLOC]

ADDQ.FOR;12                SIZE:       4/6        CREATED: 11-JUN-1981 19:08
                           OWNER:     [212,003]   REVISED: 11-JUN-1981 19:09 (1)
                           FILE ID: (1005,8,0)    EXPIRES: <NONE SPECIFIED>
  FILE PROTECTION:         SYSTEM:RWED, OWNER:RWED, GROUP:RWE, WORLD:RE
  FILE ORGANIZATION:       SEQUENTIAL
  FILE ATTRIBUTES:         ALLOCATION=6, EXTEND=0
  RECORD FORMAT:           VARIABLE LENGTH
  RECORD ATTRIBUTES:       CARRIAGE RETURN

ALLO.FOR;53                SIZE:       5/6        CREATED: 18-JUN-1981 17:32
                           OWNER:     [212,003]   REVISED: 18-JUN-1981 17:32 (1)
                           FILE ID: (1661,41,0)   EXPIRES: <NONE SPECIFIED>
  FILE PROTECTION:         SYSTEM:RWED, OWNER:RWED, GROUP:RWE, WORLD:RE
  FILE ORGANIZATION:       SEQUENTIAL
  FILE ATTRIBUTES:         ALLOCATION=6, EXTEND=0
  RECORD FORMAT:           VARIABLE LENGTH
  RECORD ATTRIBUTES:       CARRIAGE RETURN

ALLOC.FOR;111              SIZE:      25/30       CREATED: 11-JUN-1981 18:23
                           OWNER:     [212,003]   REVISED: 11-JUN-1981 18:24 (1)
                           FILE ID: (1076,7,0)    EXPIRES: <NONE SPECIFIED>
  FILE PROTECTION:         SYSTEM:RWED, OWNER:RWED, GROUP:RWE, WORLD:RE
  FILE ORGANIZATION:       SEQUENTIAL
  FILE ATTRIBUTES:         ALLOCATION=30, EXTEND=0
  RECORD FORMAT:           VARIABLE LENGTH
  RECORD ATTRIBUTES:       CARRIAGE RETURN

ALPHA.FOR;1                SIZE:       1/6        CREATED:  9-JUL-1981 15:34
                           OWNER:     [212,003]   REVISED:  9-JUL-1981 15:34 (1)
                           FILE ID: (1151,13,0)   EXPIRES: <NONE SPECIFIED>
  FILE PROTECTION:         SYSTEM:RWED, OWNER:RWED, GROUP:RWED, WORLD:RE
  FILE ORGANIZATION:       SEQUENTIAL
  FILE ATTRIBUTES:         ALLOCATION=6, EXTEND=0
  RECORD FORMAT:           VARIABLE LENGTH
  RECORD ATTRIBUTES:       CARRIAGE RETURN

AVAIL.FOR;5                SIZE:       1/6        CREATED: 26-MAY-1981 13:36
                           OWNER:     [212,003]   REVISED: 26-MAY-1981 13:36 (1)
                           FILE ID: (325,8,0)     EXPIRES: <NONE SPECIFIED>
  FILE PROTECTION:         SYSTEM:RWED, OWNER:RWED, GROUP:RWE, WORLD:RE
  FILE ORGANIZATION:       SEQUENTIAL
  FILE ATTRIBUTES:         ALLOCATION=6, EXTEND=0
  RECORD FORMAT:           VARIABLE LENGTH
  RECORD ATTRIBUTES:       CARRIAGE RETURN

BLDFIL.FOR;2               SIZE:       1/3        CREATED: 15-APR-1981 13:48
                           OWNER:     [212,003]   REVISED: 15-APR-1981 14:03 (1)
                           FILE ID: (3661,2,0)    EXPIRES: <NONE SPECIFIED>
  FILE PROTECTION:         SYSTEM:RWED, OWNER:RWED, GROUP:RWE, WORLD:RE
  FILE ORGANIZATION:       SEQUENTIAL
  FILE ATTRIBUTES:         ALLOCATION=3, EXTEND=0
  RECORD FORMAT:           VARIABLE LENGTH
  RECORD ATTRIBUTES:       CARRIAGE RETURN

CKNAME.FOR;28              SIZE:       8/12       CREATED: 11-JUN-1981 19:00
                           OWNER:     [212,003]   REVISED: 11-JUN-1981 19:00 (1)
                           FILE ID: (931,14,0)    EXPIRES: <NONE SPECIFIED>
  FILE PROTECTION:         SYSTEM:RWED, OWNER:RWED, GROUP:RWE, WORLD:RE
```

Figure 3-8. Sample Full Directory Listing

```
DUMP OF FILE: _DBB1:[FDYN.FDY03.PARM]PARTST.EXE;9 ON:  9-SEP-1981 19:05:16.28

************************  FORMATTED FILE HEADER  *************************

FILE NUMBER:                    3947        0F6B
FILE SEQUENCE:                  36          0024
RELATIVE VOLUME NUMBER:         0           0000
FILE HEADER CHECKSUM:           6470
FILE OWNER:                     [000212,000003]
STRUCTURE LEVEL:                ODS-2

FILE EXTENSION INFORMATION:
FILE NUMBER:                    0           0000
FILE SEQUENCE:                  0           0000
RELATIVE VOLUME NUMBER:         0           0000

DIRECTORY BACKLINK INFORMATION:
FILE NUMBER:                    7005        1B5D
FILE SEQUENCE:                  19          0013
RELATIVE VOLUME NUMBER:         0           0000

FILE SIZE:
END OF FILE BLOCK:              82      00000052
ALLOCATED SIZE:                 84      00000054
FIRST FREE BYTE:                0           0000

CREATION DATE:                   9-SEP-1981 18:55:36.95
REVISION DATE:                   9-SEP-1981 18:55:39.52
EXPIRATION DATE:                <NONE SPECIFIED>

FILE PROTECTION:                SYSTEM:RWED, OWNER:RWED, GROUP:RWE, WORLD:RE

FILE CHARACTERISTICS:           CONTIGUOUS-BEST-TRY

FILE EXTENT(S):
STARTING LOGICAL BLOCK NUMBER:                  COUNT:

(1)     257502.         0003EDDE                83.       00000053

***************************************************************************
```

Figure 3-9. System File Analyzer Output

3.12 FORTRAN STATIC SOURCE CODE ANALYZER PROGRAM (SAP)

The FORTRAN Static Source Code Analyzer Program (SAP) automatically produces statistics on occurrences of statements and structures within a FORTRAN program (see Reference 13.) Statistics, as well as figures of complexity, are gathered on a module-by-module basis. The SE data which might be obtained through this source are summarized in Figure 3-10. A sample of the output from SAP is shown in Figure 3-11.

MODULE TYPE AND EXTERNAL COMMUNICATION

- Module type (main, subroutine, function, or block data)
- Number of entry points
- Number of COMMON blocks referenced
- Number of names in argument list
- Number of subroutine calls
- Number of subroutine names referenced
- Number of functions called
- Number of function name referenced
- Number of external names defined
- Number of externally defined modules referenced
- Number of arithmetic statement functions (ASFs) defined
- Number of references to ASFs
- Maximum and average length of argument lists in references to subroutines and functions

COMMENTING OF MODULE

- Total number of lines of source code
- Total number of comment lines
- Total number of noncomment lines
- Length of prologue
- Number of embedded comments (total/prologue)
- Number of comments appearing after !
- Number of blank comment lines
- Maximum and average length of nonprologue comment blocks
- Maximum and average number of lines between comments

STATEMENT BREAKDOWN

- Total number of noncomment statements
- Number and percentage of executable statements
- Number and percentage of nonexecutable statements

Figure 3-10. Statistics From the FORTRAN Static Source Code Analyzer Program (SAP) (1 of 6)

STATEMENT BREAKDOWN (Cont'd)

- Number and percentage of assignment statements*
- Number and percentage of control statements*
- Number and percentage of I/O statements*
- Number and percentage of format statements*
- Number and percentage of NAMELIST statements*
- Number and percentage of data statements*
- Number and percentage of specification statements*
- Number and percentage of statement function definitions*
- Number and percentage of subprogram statements*
- Number and percentage of other statements
- Number and percentage of undefined statements**

---

*As defined by IBM GC28-6515-9, IBM S/360 and S/370 FORTRAN-IV language

**Statements not decodable by SAP

CONTROL STATEMENT BREAKDOWN

- Number of IF statements:
  - Number of logical IF statements
  - Number of arighmetic IF statements
- Number of GO TO statements:
  - Number of unconditional GO TO statements
  - Number of GO TO statements as object of IF statement
  - Number of assigned GO TO statements
  - Number of computed GO TO statements
  - Number of different labels used as targets of GO TO statements
- Number of DO statements
- Number of ERR= constructs
- Number of END= constructs
- Number of RETURN statements:
  - Number of normal RETURN statements
  - Number of RETURN i statements
- Number of PAUSE statements
- Number of STOP statements

Figure 3-10. Statistics From the FORTRAN Static Source Code Analyzer Program (SAP) (2 of 6)

CONTROL STATEMENT BREAKDOWN (Cont'd)

- Total number of branches in the code
- Number of unconditional upward transfers
- Number of nonFORMAT statements labeled
- Number of branches to label specified in an argument list
- Maximum and average level of DO loop nesting
- Maximum and average number of statements in a DO loop

ASSIGNMENT STATEMENT BREAKDOWN

- Number of assignment statements
- Maximum and average number of variables per statement
- Maximum and average number of operators per statement

SPECIFICATION STATEMENT BREAKDOWN

- Total number of variables named in module
- Number of variables referenced in executable statements
- Number of variable names referenced in COMMON statements
- Number of variable names referenced in EQUIVALENCE statements
- Maximum and average number of dimensions for arrays
- Maximum and average number of characters in variable name

SUBSCRIPT COMPLEXITY

- Maximum and average subscript complexity (i.e., number of operators and parentheses)

MODULE TYPE STATISTICS (GLOBAL)

- Total number of modules
- Number of main programs
- Number of subroutines
- Number of function modules
- Number of block data modules

Figure 3-10. Statistics From the FORTRAN Static Source Code Analyzer Program (SAP) (3 of 6)

MODULE LENGTH AND COMMENTING STATISTICS (GLOBAL)

- Total number of source lines
- Maximum and average number of source lines per module
- Total number of coded source lines
- Maximum and average number of coded source lines per module
- Total number of comment lines
- Maximum and average number of comment lines per module
- Maximum and average length of prologue
- Maximum and average number of embedded comments
- Maximum and average number of inline comments
- Maximum and average number of blank comment lines
- Maximum and average number of coded lines between comments

MODULE COMMUNICATIONS (GLOBAL)

- Total number of entry points
- Maximum and average number of entry points per module
- Total number of subroutine calls
- Maximum and average number of subroutine calls
- Total number of function calls
- Maximum and average number of function calls
- Maximum and average number of external names defined
- Maximum and average number of externally defined modules referenced
- Maximum and average number of arithmetic statement functions (ASFs) defined
- Maximum and average number of references to ASFs
- Maximum and average length of argument lists in references to subroutines and functions

STATEMENT BREAKDOWN (GLOBAL)

- Total number of noncomment statements
- Number and percentage of executable statements

Figure 3-10. Statistics From the FORTRAN Static Source Code Analyzer Program (SAP) (4 of 5)

STATEMENT BREAKDOWN (GLOBAL) (Cont'd)

- Number and percentage of nonexecutable statements
- Number and percentage of assignment statements*
- Number and percentage of control statements*
- Number and percentage of I/O statements*
- Number and percentage of format statements*
- Number and percentage of NAMELIST statements*
- Number and percentage of data statements*
- Number and percentage of specifications statements*
- Number and percentage of statement function definitions*
- Number and percentage of subprogram statements
- Number and percentage of other statements
- Number and percentage of undecoded statements**

---

*As defined by IBM G28-6515-9, IBM S/360 and S/370 FORTRAN-IV language
**Statements not decodable by SAP

CONTROL STATEMENT BREAKDOWN (GLOBAL)

- Maximum and average number of IF statements per module
- Maximum and average number of GO TO statements per module
- Maximum and average number of DO statements per module
- Maximum and average level of DO loop nesting
- Maximum and average number of statements per DO loop

ASSIGNMENT STATEMENT BREAKDOWN (GLOBAL)

- Number of assignment statements
- Maximum and average number of variables per statement
- Maximum and average number of operators per statement

Figure 3-10. Statistics From the FORTRAN Static Source Code Analyzer Program (SAP) (5 of 6)

SPECIFICATION STATEMENT BREAKDOWN (GLOBAL)

- Maximum and average number of variables named per module
- Maximum and average number of variables referenced in executable statements per module
- Maximum and average number of variable names referenced in COMMON statements per module
- Maximum and average number of variable names referenced in EQUIVALENCE statements per module
- Maximum and average number of dimensions per array
- Maximum and average number of characters in a variable name

SINGLE STATEMENT COMPLEXITY

- Maximum and average subscript complexity (i.e., number of operators and parentheses)

Figure 3-10. Statistics From the FORTRAN Static Source Code Analyzer Program (SAP) (6 of 6)

```
11:13:59 03-DEC-80 SOURCE ANALYZER PROGRAM       MODULE STATISTICS SUMMARY FILE          LUN  7                 PAGE  1
                                           MODULE TYPE = MAIN PROGRAM    MODULE NAME =

                                                EXTERNAL COMMUNICATIONS
NUMBER OF ENTRY POINTS                        0          NUMBER OF RETURN STATEMENTS                              0
NUMBER OF ARGUMENTS TO MODULE                 0          NUMBER OF NAMED COMMON BLOCKS REFERENCED                 0
NUMBER OF SUBROUTINE CALLS                   11          NUMBER OF SUBROUTINE NAMES REFERENCED                    9
NUMBER OF FUNCTION REFERENCES                18          NUMBER OF FUNCTION NAMES REFERENCED                      1
NUMBER OF EXTERNAL NAMES DEFINED              0          NUMBER OF REFERENCES TO EXTERNALLY DEFINED NAMES         0
NUMBER OF ARITH. STATEMENT FUNCTIONS DEFINED  0          NUMBER OF REFERENCES TO ARITH. STATE. FUNCTIONS          0
MAXIMUM AND AVERAGE LENGTH OF ARGUMENT LISTS IN REFERENCES TO SUBROUTINES AND FUNCTIONS      11   2.0

                                                   MODULE COMMENTING
NUMBER OF SOURCE LINES                       75          NUMBER OF CODED LINES                                   55
NUMBER OF COMMENT LINES                      20          NUMBER OF PROLOGUE LINES                                 9
NUMBER OF EMBEDDED COMMENT LINES             11          NUMBER OF IN-LINE COMMENTS                               0
NUMBER OF BLANK COMMENT CARDS                10
MAXIMUM AND AVERAGE LENGTH OF NON-PROLOGUE COMMENT PACKETS                  4   1.8
MAXIMUM AND AVERAGE NUMBER OF LINES BETWEEN COMMENT PACKETS                13   6.9

                                                STATEMENT CLASS COUNTERS

NUMBER AND PERCENTAGE OF EXECUTABLE STATEMENTS    31  68.9   NUMBER AND PERCENTAGE OF NON EXECUTABLE STATEMENTS  14  31.1
NUMBER AND PERCENTAGE OF ASSIGNMENT STATEMENTS     4   8.9   NUMBER AND PERCENTAGE OF CONTROL STATEMENTS         20  44.4
NUMBER AND PERCENTAGE OF SUBPROGRAM STATEMENTS     1   2.2   NUMBER AND PERCENTAGE OF SPEC. STATEMENTS            3   6.7
NUMBER AND PERCENTAGE OF TYPE SPEC. STATEMENTS     2   4.4   NUMBER AND PERCENTAGE OF DATA STATEMENTS             0   0.0
NUMBER AND PERCENTAGE OF I/O STATEMENTS            7  15.6   NUMBER AND PERCENTAGE OF FORMAT STATEMENTS           3   6.7
NUMBER AND PERCENTAGE OF NAMELIST STATEMENTS       0   0.0   NUMBER AND PERCENTAGE OF STRUCTURE STATEMENTS        0   0.0
NUMBER AND PERCENTAGE OF OTHER STATEMENTS          5  11.1   NUMBER AND PERCENTAGE OF UNDECODED STATEMENTS        0   0.0
NUMBER AND PERCENTAGE OF ARITHMETIC STATEMENT FUNCTION DEFINITIONS      0   0.0

                                                   STATEMENT TYPE COUNTERS
NUMBER OF ASF DEF.   STATEMENTS =   0.   NUMBER OF ASSIGNMENT STATEMENTS =   4.   NUMBER OF ACCEPT     STATEMENTS =   0.
NUMBER OF ASSIGN     STATEMENTS =   0.   NUMBER OF BACKSPACE  STATEMENTS =   0.   NUMBER OF BLOCKDATA  STATEMENTS =   0.
NUMBER OF BYTE       STATEMENTS =   0.   NUMBER OF CALL       STATEMENTS =  11.   NUMBER OF CLOSE      STATEMENTS =   2.
NUMBER OF COMMON     STATEMENTS =   1.   NUMBER OF COMPLEX    STATEMENTS =   0.   NUMBER OF CONTINUE   STATEMENTS =   0.
NUMBER OF DATA       STATEMENTS =   0.   NUMBER OF DECODE     STATEMENTS =   0.   NUMBER OF DEFINEFILE STATEMENTS =   0.
NUMBER OF DIMENSION  STATEMENTS =   1.   NUMBER OF DOUBLEPREC STATEMENTS =   0.   NUMBER OF DOWHILE    STATEMENTS =   0.
NUMBER OF DO         STATEMENTS =   0.   NUMBER OF ELSE       STATEMENTS =   0.   NUMBER OF ENCODE     STATEMENTS =   0.
NUMBER OF ENDDO      STATEMENTS =   0.   NUMBER OF ENDFILE    STATEMENTS =   0.   NUMBER OF ENDIF      STATEMENTS =   0.
NUMBER OF END        STATEMENTS =   1.   NUMBER OF ENTRY      STATEMENTS =   0.   NUMBER OF EQUIVALENC STATEMENTS =   1.
NUMBER OF EXTERNAL   STATEMENTS =   0.   NUMBER OF FIND       STATEMENTS =   0.   NUMBER OF FORMAT     STATEMENTS =   3.
NUMBER OF FUNCTION   STATEMENTS =   0.   NUMBER OF GOTO       STATEMENTS =   8.   NUMBER OF .IF        STATEMENTS =   0.
NUMBER OF IF         STATEMENTS =   1.   NUMBER OF IMPLICIT   STATEMENTS =   0.   NUMBER OF INCLUDE    STATEMENTS =   5.
NUMBER OF INTEGER    STATEMENTS =   0.   NUMBER OF LOGICAL    STATEMENTS =   1.   NUMBER OF NAMELIST   STATEMENTS =   0.
NUMBER OF OPEN       STATEMENTS =   2.   NUMBER OF PARAMETER  STATEMENTS =   0.   NUMBER OF PAUSE      STATEMENTS =   0.
NUMBER OF PRINT      STATEMENTS =   0.   NUMBER OF PROGRAM    STATEMENTS =   0.   NUMBER OF READ       STATEMENTS =   0.
NUMBER OF REAL       STATEMENTS =   1.   NUMBER OF RETURN     STATEMENTS =   0.   NUMBER OF REWIND     STATEMENTS =   0.
NUMBER OF STOP       STATEMENTS =   0.   NUMBER OF SUBROUTINE STATEMENTS =   0.   NUMBER OF TYPE       STATEMENTS =   0.
NUMBER OF WRITE      STATEMENTS =   3.   NUMBER OF UNDECODED  STATEMENTS =   0.   NUMBER OF
```

Figure 3-11. Sample Output From the FORTRAN Static Source Code Analyzer Program (SAP) (1 of 2)

```
11:14:00 03-DEC-80 SOURCE ANALYZER PROGRAM       MODULE STATISTICS SUMMARY FILE          LUN 7                    PAGE  2

                                                CONTROL STATEMENT BREAKDOWN
TOTAL NUMBER OF BRANCHES                           11          TOTAL NUMBER OF IF STATEMENTS                         1
NUMBER OF LOGICAL IF STATEMENTS                     1          NUMBER OF ARITHMETIC IF STATEMENTS                    0
TOTAL NUMBER OF GOTO STATEMENTS                     8          NUMBER OF UNCONDITIONAL GOTO STATEMENTS               7
NUMBER OF UNCONDITIONAL UPWARD TRANSFERS            7          NUMBER OF UNCONDITIONAL DOWNWARD TRANSFERS            0
NUMBER OF ASSIGNED GOTO STATEMENTS                  0          NUMBER OF COMPUTED GO TO STATEMENTS                   1
NUMBER OF GOTO'S AS OBJECTS OF IF'S                 0          NUMBER OF TARGET LABELS                               5
NUMBER OF ERR= CONSTRUCTS                           2          NUMBER OF END= CONSTRUCTS                             0
NUMBER OF RETURN STATEMENTS                         0          NUMBER OF RETURN 'I' STATEMENTS                       0
NUMBER OF PAUSE STATEMENTS                          0          NUMBER OF STOP STATEMENTS                             0
NUMBER OF DO STATEMENTS                             0
MAXIMUM AND AVERAGE DEPTH OF DO LOOP NESTING                                      0    0.0
MAXIMUM AND AVERAGE NUMBER OF STATEMENTS/LOOP                                     0    0.0

                                                ASSIGNMENT STATEMENT BREAKDOWN
TOTAL NUMBER OF ASSIGNMENT STATEMENTS               4
MAXIMUM AND AVERAGE NUMBER OF VARIABLES PER STATEMENT                             2   1.2
MAXIMUM AND AVERAGE NUMBER OF OPERATORS PER STATEMENT                             0   0.0

                                                SPECIFICATION STATEMENT BREAKDOWN
TOTAL NUMBER OF VARIABLES NAMED IN MODULE          38          NUMBER OF VARIABLES REFERENCED IN CODE               35
NUMBER OF VARIABLES NAMED IN COMMON BLOCKS          1          NUMBER OF COMMON BLOCK VARIABLES REFERENCED           1
NUMBER OF VARIABLES EQUIVALENCED                    2
MAXIMUM AND AVERAGE NUMBER OF DIMENSIONS FOR ARRAYS                               1   1.0
MAXIMUM AND AVERAGE NUMBER OF CHARACTERS IN A VARIABLE NAME                       6   3.6

                                                SUBSCRIPT COMPLEXITY
MAXIMUM AND AVERAGE SUBSCRIPT COMPLEXITY                                          1   1.0

                                                HALSTEAD OPERATORS AND USE COUNT
```

Figure 3-11. Sample Output From the FORTRAN Static Source Code Analyzer Program (SAP) (2 of 2)

SECTION 4 - SEL DATA THAT CANNOT BE EXTRACTED AUTOMATICALLY

Not all of the efforts expended during software development can be accounted for via automated data collection. This is primarily due to the fact that these efforts cannot be quantified or measured in any precise way. For example, during the implementation of even some of the simplest algorithms, false starts frequently can be made before a workable solution is found (i.e., much of what is done is by trial and error). Also, portions of a design may lend themselves to easy solution, while others, because of constraints imposed by the project or mission, may be very difficult to define. The effort expended on these kinds of activities is not readily available for measure.

This section lists and discusses some of the items of data currently collected via the SEL software engineering forms which cannot be collected automatically. Table 4-1 summarizes these types of data. The data generally fall into the following categories:

- Subjective data
- Manual processes
- Valid other activities

4.1 SUBJECTIVE DATA

Much of the data collected from the SEL forms is subjective in nature. For example, what constitutes a "good" run depends on each individual's interpretation of what "good" means. Another example is the use of the word "simple" to describe software complexity. Those who understand a section of software will tend to call the section "simple," whereas those who do not understand it may well call it complex.

Table 4-1. Data From the SEL Forms That Cannot Be Automatically Extracted (1 of 3)

SEL Form	Data Item
Resource Summary	Manpower hours*
	Other charges
	Percent of management
Run Analysis	Run purpose
Change Report	Reason for change
	Effect
	Effort
	Type of change
	Code reading
	Activities used for program validation
	Activities successful in detecting error symptoms
	Activities tried to find cause
	Activities successful in finding cause
	Time required to isolate the cause
	When did error enter the system
Component Status Report	Formal review
	Design walk-through
	Critical design reviews
	Code reading
	Valid other activities ($$xxxxxx indicates form entry name):
	Acceptance testing
	Filling out the SEL forms
	Meetings
	Training
	Travel (to and from GSFC)

*Manpower hours might be obtained in the form of tapes such as those used in the Manpower Allocation and Reporting System (MARS) (Reference 14) or the Financial Reporting (FINREP) Program (Reference 15).

Table 4-1. Data From the SEL Forms That Cannot Be Automatically Extracted (2 of 3)

SEL Form	Data Item
Component Status Report (Cont'd)	Valid other activities (Cont'd):
	JCL development time
	Overlay development time
	System description development time
	User's guide development time
	Discussion with analysis personnel ($$ANALYT)
	Block time ($$BLKTIM)
	Discussion with other development personnel ($$CONSUL)
	Data generation ($$DATGEN)
	Data set formats and maintanence ($$DATSET)
	Demonstrations ($$DEMO)
	Preparation of task implementation plan ($$IMPLAN)
	Discussion with task personnel ($$INTERF)
	Keypunching ($$KEYPCH)
	Review GESS, IBM, or other manual ($$MANUAL)
	Write formal memoranda ($$MEMO)
	Monthly Progress Report preparation ($$MNTHLY)
	Design notebook preparation ($$NOTEBK)
	Informal memos/instruction preparation ($$PAPERW)
	Planning (not milestones) ($$PLANS)
	Preparation for presentation ($$PRESNT)
	Work on questions ($$QUESTS)
	Review old software ($$ROSW)

Table 4-1. Data From the SEL Forms That Cannot Be Automatically Extracted (3 of 3)

SEL Form	Data Item
Component Status Report (Cont'd)	Valid other activities (Cont'd):
	Review requirements/specifications for design ($$RREQS)
	Review standards/methodology ($$RSTDS)
	Prepare schedules (milestones) ($$SCHEDL)
	Attend seminar ($$SEMINR)
	Simulation support ($$SIM)
	Status meeting with management ($$STATUS)
	Generate system tape ($$SYSTAP)
	Perform system testing ($$SYSTST)
	Write test plan ($$TESTPL)
	Work on tool (not part of system) ($$TOOL)
	Weekly Progress Report ($$WEEKLY)
	Xeroxing (reproduction) ($$XEROX)
Project Summary Report	Complexity (hard, easy, moderate)

These types of subjective conflicts point out the need for better metrics by which to quantify and qualify the data being collected. Given a measure of complexity expressed in terms of simple structured properties (such as the number or interactions between product and organizational elements), normalized measures for programming effort, systems reliability, productivity, and security can be devised, and meaningful comparisons between different products or methodologies can be made. Without such measures, may of the essential parts of the developing discipline remain unconnected and easily misunderstood. Success in developing metrics will provide a much needed measure of consistency in the results obtained (see Reference 16).

4.2 MANUAL PROCESSES

Another important consideration is that certain aspects of current software development are inherently manual or non-automated processes. The following are examples of such manual processes: design reviews, code reading, and meetings. Activities such as these are categorically outside of the realm of automation.

4.3 VALID OTHER ACTIVITIES

Items which are generally categorized as "valid other activities" (for the Component Status Report) also are not amenable to automation. These include activities such as travel, review of old software, review of design requirements, etc. (see the data items for the Component Status Report in Table 4-1). However, these activities have a direct bearing and impact on the costs and the success or failure of software development projects, and they cannot be ignored.

SECTION 5 - FUNCTIONAL REQUIREMENTS

This section gives some top-level functional requirements for an online automated data collection system. Both operational considerations and the SEL hardware environment are factors in these requirements.

5.1 OPERATIONAL CONSIDERATIONS

If the data collection system is to accurately measure the true activity of the software development process, the act of collecting data must not significantly interfere with development activities. Also, the performance of the operating system as a whole must not be degraded by the data collector. With this in mind, the major design goals of the data collector are the following:

- Transparency--The user should not be aware that he is being monitored or that data are being collected.
- Efficiency--Both time and space utilized must be optimized.

The efficient use of time and space and the event monitoring by the automated data collection system is discussed in the following subsections.

5.1.1 TIME AND SPACE UTILIZATION

In general, there will be many events that will be monitored; therefore, the time spent logging each event must be minimal. Only the essential data should be collected, and it should be possible to selectively monitor development projects. Also, the data collection manager or system programmer must be able to easily turn the collector on and off.

The space taken up by the data collector will have to be minimized. It would not be feasible to develop a monitor that would be so large that it wouldn't fit into core along with the application it is to measure.

Taking these factors into account, the SEL data collector must be designed to take the significant information about an event (e.g., its type, the time, data unique to the event) and store it for subsequent analysis. Since some events will have more data associated with them than do others, the records of the intermediate storage file should be variable in length in order to conserve storage space.

5.1.2 EVENT MONITORING

The data collector must be capable of monitoring three classes of events: resource use, logical interrupts, and flow of control. The specific items monitored will vary, depending on the software development phase (e.g., requirements, design, coding which is active for a given project.

The resources utilized by a user are perhaps the most easily collectible items, since they are generally available in some form through system accounting and resource utilization procedures. Items such as CPU time, core usage, page frame allocation and faulting, disk usage, I/O interrupts, etc., need only be extracted and stored.

However, routines that normally service an event must be capable of calculating many of the other items of interest directly or must call existing or newly developed software engineering tools capable of deriving more detailed statistics from some basic input source. Programs such as the FORTRAN Static Source Code Analyzer Program (SAP) and the Multi-Level Expression Design Language - Requirements (MEDL-R) (briefly discussed in Sections 3.11 and 3.4, respectively) are representatives of this class of tools.

5.2 DATA COLLECTION IN THE SEL HARDWARE ENVIRONMENT

The SEL is a complex system environment in which a telecommunications network is attached to a computing complex consisting of a DEC PDP-11/70 and VAX-11/780. The computing environment is under control of the VAX/VMS and RSX-11M operating systems. User-written application programs execute upon demand from local and remote interactive terminals. Batch processing can also be performed concurrently.

Automated data collection in this environment requires both a definition of purpose and a methodology which can be used to accomplish that purpose. Considerations include the overall SEL hardware environment, system performance, computing workload, and transmission speed. Because of core limitations on the PDP-11/70, space requirements in memory and on disk are key constraints on the approach taken to automated data collection.

The computers in the SEL environment, although developed by the same manufacturer, have very distinct operating characteristics and systems. Consequently, it may be necessary to take entirely different approaches to data collection on the two machines. This would be less desirable, however, than a centralized data collection facility which would be shareable between the computers through a network such as DECnet (Reference 17). A network of this type would permit synchronization of the system clocks and enable concurrent data collection on the two machines with a single executive controller. This feature is important because it would minimize the amount of preprocessing of intermediate records prior to their entry into the SEL data base.

5.3 SUMMARY

In developing a software engineering data collection system, certain general requirements regarding the data collection environment become evident. These are summarized below.

1. The act of collecting data must be transparent to project being monitored.

2. The act of establishing and activating data collection interfaces must be capable of being dynamic (as well as static) and of being performed on any ongoing process without logically interrupting that process.

3. The data collection system must support the definition of event discriptors whose content defines the conditions under which a recording of data is to be made for later analysis. Such a descriptor might contain the following:

   a. Time
   b. Project
   c. Data and values
   d. Level of collection

4. The data collection function must not be subject to being disabled for that period of time for which data collection is required for a given project.

5. The data collection system must support the acts of event detection and recording of the captured data. In a data-rich environment, the sharing of a physical resource must be transparent to an application program (process).

6. The level of system support for the data collector must be standardized across application systems and across hardware/software systems (e.g., VAX, PDP, IBM S/360).

7. The data collection terminology must be standardized throughout the data collection environment.

8. The ability to logically save the most recently recorded data prior to any purging of the data by another process or subprocess in the system is necessary.

9. Data identification must be provided to distinguish data between projects. The identification of a collected item must be monitored as part of the data collection function.

10. There must be compatibility with the current SEL data base. The automated data collection should be considered to be an adjunct to the established data base mechanism. The format of data collected must be designed so that existing data base formats continue to be satisfactory.

11. The data collector must be able to monitor both batch and interactive processes.

12. Because of the high volume of data collected in an automated environment, procedures for maintaining the collected data prior to integration into the SEL data base must be established.

13. The ability to edit/purge selected portions of the collected data must be provided.

14. Time tagging of data across projects is desirable if the chronology of the collected data is of interest. If data are time tagged, it will then be possible to develop a decay function so that the most recent data is not lost. This is essential if intermediate storage for collected data is in short supply.

15. Shared access by multiple processes of the intermediate collection file(s) is essential, since it is likely that several users for a given project will be active concurrently. It may be necessary to synchronize the accessibility to project files (enqueue/dequeue).

SECTION 6 - CONCLUSIONS AND RECOMMENDATIONS

Currently, the process of large-scale program development and maintenance in the SEL is informal. Its costs are high and its output is variable. However, it is essential to study the process as it is evolving and to make organized, quantized records of observations which familiarize the perception of what is occurring. With such global statistics (over the entire life cycle), it is hoped that specific points or sources of trouble can be identified. Perhaps areas of the development process which can be better understood can also be identified. Only then can an attempt be made to change the process without the risk of achieving only local optimization.

In order to automatically collect statistics on software development, it is essential that a higher degree of automated software development tools be developed which support the entire software life cycle. Further, it is necessary that formal software development procedures be established and applied routinely to development efforts. The programmer workbench (discussed in Section 3.5) is a major step in this area. Once formalized, the procedures become easier to automate, and, therefore, data collection for all development phases can be realized.

It is recommended that work be started to define and develop tools which support the entire development life cycle. Special attention should be given to the design phase, which is by far the most difficult to represent in a computer and is therefore the most difficult to automate. It is further recommended that SEL-enriched accounting software be developed and coupled with revised software engineering forms which address the desired subjective data.

It is not currently possible to automatically collect statistics on all areas of software development, but much of the overhead and cost related to data collection can be reduced. By integrating data collection with a system which supports the entire development process, more data of a higher quality can be collected. It is hoped that this will provide a clearer insight on how to develop quality software.

APPENDIX A - SAMPLE SEL SOFTWARE ENGINEERING FORMS

This appendix provides examples of the software engineering forms currently in use in the SEL. They are given in the following order:

1. General Project Summary form
2. Component Summary form
3. Resource Summary form
4. Component Status Report form
5. Computer Program Run Analysis form
6. Change Report form

GENERAL PROJECT SUMMARY

PROJECT NAME ______________________ DATE __________

A. PROJECT DESCRIPTION

Description ______________________

Form of Input ______________________

Requirements ______________________

Products Developed ______________________

Products Delivered ______________________

B. RESOURCES

Target Computer Systems __________ Development Computer Systems __________

Constraints: Execution Time __________ Size __________

Other __________

Any Problems in Meeting Constraints? __________

Useful Items from Similar Projects:

Project	Specification				Design				Code			
	%	Major	Minor	None	%	Major	Minor	None	%	Major	Minor	None

C. TIME

Start Date ______ End Date ______ Estimated Lifetime ______ Mission Date ______

Confidence Level ______

D. Cost

Cost $ ______ Maximum Available $ ______ Confidence Level ______

How Cost Determined ______________________

Personnel: Inception ______ 1/3 Way ______ 2/3 Way ______ Completion ______

Total Person Months ______

Other Costs: Computer Time ______ (hrs) Documentation $ ______

Other ( ) ______ Other ( ) ______

E. SIZE

Size of System ______ Words. ______ Data Words ______ Instructions

Maximum Space Available ______ Words. Confidence Level ______

Total Number of Source Statements: FORTRAN ______ ALC ______

Other ( ) ______

Structure of System (Check One):

___ Single Overlay

___ Overlay Structure (Number of Overlays ______ Avg. Size ______)

___ Independent Programs (Number of Programs ______ Avg. Size ______)

Define Your Concept of a Module ______________________

Number of Modules ______ Range in Module Size: Min. ______ Max. ______ Avg. ______

Number of Different I/O Formats ______

580-1 (2/77)

Figure A-1. General Project Summary Form (1 of 5)

F. COMPUTER ACCESS (Check All That Apply. Who Has Access to What.)

	Librarian	Programmer
Keying in New Source Code		
Keying in Update of Source Code		
Inclusion of Code Into System		
Submitting Compilations		
Module Testing		
Integration Testing		
Utility Runs (Tape Backup, Etc.)		

Give Percentages for Types of Access:

	Librarian	Programmer
% Batch		
% Interactive		

G. TECHNIQUES EMPLOYED (Check All That Apply and Give Level at Which Used.)

Specification:	Used	Level		Used	Level
Functional			Procedural		
English			Formal		

Design:

Top Down			Bottom Up		
Iterative Enhance.			Hardest First		
Other:			None Used		

Development:

Top Down			Bottom Up		
Iterative Enhance.			Hardest First		
Other:			None Used		

Coding:

Simulating Construct			Structured Code		
Other:			None		

Validation/Verification: Testing

Top Down (Stubs)			Bottom Up (Drivers)		
Other:			Specification Driven		
Structure Driven			None		

Validation/Verification: Inspection

Code Reading			Walk Through		
Proof:			None		

H. FORMALISMS USED

	Used	Level	Phases
PDL			
HIPO			
Flowcharts			
Baseline Diag. (Tree Ch.)			
HOS			
Functions			
Other:			
Other:			

580-1 (2/77) Continuation

Figure A-1. General Project Summary Form (2 of 5)

I. AUTOMATED TOOLS USED

Name	Phases in Which Used	Level

J. ORGANIZATION

How are the Personnel Organized: ____________________

____

Project Personnel:

Title	Job Description	Number	Names and Affiliations (If Known)

K. STANDARDS

Type ____________ Optional ______ Required ______
Title of Document ____________________

Type ____________ Optional ______ Required ______
Title of Document ____________________

Type ____________ Optional ______ Required ______
Title of Document ____________________

Type ____________ Optional ______ Required ______
Title of Document ____________________

Type ____________ Optional ______ Required ______
Title of Document ____________________

Type ____________ Optional ______ Required ______
Title of Document ____________________

Type ____________ Optional ______ Required ______
Title of Document ____________________

Type ____________ Optional ______ Required ______
Title of Document ____________________

580-1 (2/77) Continuation

Figure A-1. General Project Summary Form (3 of 5)

L. MILESTONES

Phase ______ Estimated Date ______ Confidence Level ______
How Determined ______
Reviewers ______
Reporting Procedure ______
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ______ Estimated Date ______ Confidence Level ______
How Determined ______
Reviewers ______
Reporting Procedure ______
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ______ Estimated Date ______ Confidence Level ______
How Determined ______
Reviewers ______
Reporting Procedure ______
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ______ Estimated Date ______ Confidence Level ______
How Determined ______
Reviewers ______
Reporting Procedure ______
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ______ Estimated Date ______ Confidence Level ______
How Determined ______
Reviewers ______
Reporting Procedure ______
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ______ Estimated Date ______ Confidence Level ______
How Determined ______
Reviewers ______
Reporting Procedure ______
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ______ Estimated Date ______ Confidence Level ______
How Determined ______
Reviewers ______
Reporting Procedure ______
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ______ Estimated Date ______ Confidence Level ______
How Determined ______
Reviewers ______
Reporting Procedure ______
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

580-1 (2/77) Continuation

Figure A-1. General Project Summary Form (4 of 5)

M. DOCUMENTATION

Type ______ Purpose ______
Estimated Date ______ Estimated Size ______ Tools Used ______

Type ______ Purpose ______
Estimated Date ______ Estimated Size ______ Tools Used ______

Type ______ Purpose ______
Estimated Date ______ Estimated Size ______ Tools Used ______

Type ______ Purpose ______
Estimated Date ______ Estimated Size ______ Tools Used ______

Type ______ Purpose ______
Estimated Date ______ Estimated Size ______ Tools Used ______

Type ______ Purpose ______
Estimated Date ______ Estimated Size ______ Tools Used ______

Type ______ Purpose ______
Estimated Date ______ Estimated Size ______ Tools Used ______

N. PROBLEMS

State the three most difficult problems you expect to encounter in completing the project. (1 = most difficult)

1. ______

2. ______

3. ______

O. QUALITY ASSURANCE

State the three most important aspects of the design, development and testing of the system to which you attribute your confidence in the completed system. (1 = most important)

1. ______

2. ______

3. ______

PERSON FILLING OUT FORM ______

580-1 (2/77) Continuation

Figure A-1. General Project Summary Form (5 of 5)

COMPONENT SUMMARY

PROJECT__________ DATE__________

NAME OF COMPONENT__________ CREATION DATE__________

BRIEF DESCRIPTION__________

STATUS OF COMPONENT NEW______ UNDER DEVEL______ COMPLETED______

TYPE OF SOFTWARE (Check All That Apply)

____ I/O Processing
____ Algorithmic
____ Logic Control
____ Systems Related
____ DATA/COMMON Block
____ Other

A. CODE SPECIFICATIONS (Check All That Apply)

FORM OF DESIGN	LEVEL OF DETAIL				
	Component	Subcomponent	Basic Block Segment	Stmt	Other
Functional					
Procedural					
English					
Formal					
Other ( )					

Precision of Code Specification Very Precise______ Precise______ Imprecise______

B. INTERFACES

Number Components Called______ Names______
______ Not Fully Specified ______

Number Calling This Component______ Names______
______ Not Fully Specified ______

Number Shared Items______ Names______
______ Not Fully Specified ______

Number of Components Directly Descended from This Component______ Names______
______ Not Fully Specified ______

C. PROGRAMMING LANGUAGES

Languages Used and Percentages______ (____) ______ (____)

CONSTRAINT PROBLEM EXPECTED:

	Constraint Present	Component Meets Constraint
Memory Space		
Execution Time		
Other ( )		

Size: Source Statements (Including Comments) ______ Machine Bytes______
Source Statements (Not Including Comments) ______

Useful Items From Similar Projects

Component	Project	Specification				Design				Code			
		%	Major	Minor	None	%	Major	Minor	None	%	Major	Minor	None

580-5 (6/78)

Figure A-2. Component Summary Form (1 of 2)

D. COMPLEXITY

Complexity of Function Easy __________ Moderate __________ Hard __________

________ % Assignment Statements ________ % Control Statements ________ % Other Statements (e.g., Data Decl, I/O)

E. RESOURCES TO IMPLEMENT

	Runs	Computer Time (min)	Effort (hrs)	Est. Completion Date
Design				
Code				
Test				

F. Is this component independent of the existing components? ________ Yes ________ No

If No, describe relation of this component to the existing system:

_____ inserted as a lower level elaboration of higher level components (names) __________

_____ added as a driver or interface for existing components (names) __________

_____ a redesign (to add new capability) of existing components (names) __________

_____ a renaming of existing component (name) __________

_____ regrouping of existing material from several components (names) __________

_____ other __________

Type of Addition:

_____ error correction

_____ planned enhancement

_____ implementation of requirements change

_____ improvement of clarity, maintainability, or documentation

_____ other (explain below)

_____ improvement of user service

_____ utility for development purposes only

_____ optimization of time/space/accuracy

_____ adaptation to environment change

G. ADDITIONAL COMMENTS

H. PERSON RESPONSIBLE FOR IMPLEMENTING COMPONENT __________

I. PERSON FILLING OUT FORM __________

580-5 (6/78)

Figure A-2. Component Summary Form (2 of 2)

RESOURCE SUMMARY

PROJECT ________________________ DATE __________

NAME ________________________

WEEK OF:											
MANPOWER (HOURS)											% OF MGMT.
COMPUTER USAGE (NO. RUNS/HOURS CHARGED)											
OTHER CHARGES TO PROJECT											

580-3 (6/78)

Figure A-3. Resource Summary Form

COMPONENT STATUS REPORT

PROJECT ______________________________ DATE ______________

PROGRAMMER ______________________________

COMPONENT	DESIGN			CODE DEVELOPMENT			TEST			OTHER	
	CREATE	READ	FORMAL REVIEW	CODE	READ	FORMAL REVIEW	UNIT	INTEG	REVIEW	ACTIVITY	HRS
										Travel	
										Forms	
										Meetings	
										Acc Test	
										Training	
JCL											
Overlay											
User Guide											
System Desc											

580-4 (6/78)

Figure A-4. Component Status Report Form

COMPUTER PROGRAM RUN ANALYSIS

PROGRAMMER ______________ PROJECT ______________ COMPUTER ______________ DATE ___/___/___

JOB ID	DATE MM DD	INTERACTIVE	RUN PURPOSE							COMPONENTS OF INTEREST	First Run	Run Met Objectives	Run Did Not Meet Objectives	RUN RESULTS											COMMENT (e.g., Run Lost, No Results)
														Good Run	Setup Error			Machine Error		Program Error					
			Unit Test	System Test	Benchmark Test	Maintenance/Utility	Compile/Assembly/Link	Debug Run	Other					Good Run	Submit Error	JCL Error	Other Setup Error	Hardware Error	Software Error	Compile Error	Link Error	Execute Error	User Generated Msg.	Ran to Completion	

580-7 (6/78)

Figure A-5. Computer Program Run Analysis Form

NUMBER ____________

CHANGE REPORT FORM

PROJECT NAME ____________________ CURRENT DATE ____________

SECTION A - IDENTIFICATION

REASON: Why was the change made? ____________________

DESCRIPTION: What change was made? ____________________

EFFECT: What components (or documents) are changed? (Include version) ____________________

EFFORT: What additional components (or documents) were examined in determining what change was needed? ____________________

	(Month	Day	Year)
Need for change determined on			
Change started on			

What was the effort in person time required to understand and implement the change?

_____1 hour or less, _____1 hour to 1 day, _____1 day to 3 days, _____more than 3 days

SECTION B - TYPE OF CHANGE (*How is this change best characterized?*)

- ☐ Error correction
- ☐ Planned enhancement
- ☐ Implementation of requirements change
- ☐ Improvement of clarity, maintainability, or documentation
- ☐ Improvement of user services
- ☐ Insertion/deletion of debug code
- ☐ Optimization of time/space/accuracy
- ☐ Adaptation to environment change
- ☐ Other (Explain in E)

Was more than one component affected by the change? Yes _______ No _______

FOR ERROR CORRECTIONS ONLY

SECTION C - TYPE OF ERROR (How is this error best characterized?)

- ☐ Requirements incorrect or misinterpreted
- ☐ Functional specifications incorrect or misinterpreted
- ☐ Design error, involving several components
- ☐ Error in the design or implementation of a single component
- ☐ Misunderstanding of external environment, except language
- ☐ Error in use of programming language/compiler
- ☐ Clerical error
- ☐ Other (Explain in E)

FOR DESIGN OR IMPLEMENTATION ERRORS ONLY

If the error was in design or implementation:

The error was a mistaken assumption about the value or structure of data ____________

The error was a mistake in control logic or computation of an expression ____________

580-2 (6/78)

Figure A-6. Change Report Form (1 of 2)

FOR ERROR CORRECTIONS ONLY

SECTION D - VALIDATION AND REPAIR

What activities were used to validate the program, detect the error, and find its cause?

	Activities Used for Program Validation	Activities Successful in Detecting Error Symptoms	Activities Tried to Find Cause	Activities Successful in Finding Cause
Pre-acceptance test runs				
Acceptance testing				
~~Post acceptance use~~				
Inspection of output				
Code reading by programmer				
Code reading by other person				
Talks with other programmers				
Special debug code				
System error messages				
Project specific error messages				
Reading documentation				
Trace				
Dump				
Cross-reference/attribute list				
Proof technique				
Other (Explain in E)				

What was the time used to isolate the cause?

____one hour or less, ____one hour to one day, ____more than one day, ____never found

If never found, was a workaround used?________Yes________No (Explain in E)

Was this error related to a previous change?

______Yes (Change Report #/Date__________) ____No ____Can't tell

When did the error enter the system?

____requirements ____functional specs ____design ____coding and test ____other ____can't tell

SECTION E - ADDITIONAL INFORMATION

Please give any information that may be helpful in categorizing the error or change, and understanding its cause and its ramifications.

Name:________________________ Authorized:________________________ Date:__________

580-2 (6/78)

Figure A-6. Change Report Form (2 of 2)

REFERENCES

1. Basili, V. R., IEEE Computer Society, "Data Collection, Validation and Analysis," Fall 1980

2. Computer Sciences Corporation, CSC/TM-81/6104, The Software Engineering Laboratory (Preliminary), D. N. Card, September 1981

3. Teichrow, D. and E. A. Hershey, Classics in Software Engineering, E. N. Yourdon (editor). New York: Yourdon Press, 1979, pp. 389-407

4. Computer Sciences Corporation, CSC/TM-80/6093, Multi-Level Expression Design Language - Requirements (MEDL-R) System Evaluation, W. Decker, May 1980

5. University of Michigan (ISDOS Project, Department of Industrial and Operations Engineering), ESD-TR-78-127, Vol. I, User's Requirements Language (URL) User's Manual: Part I - Description, H6180/Multics/Version 3.2, D. Teichrow, E. A. Hershy, and S. Spevak, March 1977

6. --, ESD-TR-78-129, Vol. II, User's Requirements Language (URL) User's Manual: Part II - References, IBM/370/MVS/TSO/Version 3.2, D. Teichrow, E. A. Hershy, and S. Spevak, March 1977

7. --, ESD-TR-78-130, Vol. I, User's Requirements Language (URL) User's Manual: Part I - Description, H1680/Multics/Version 3.3, D. Teichrow, E. A. Hershy, and S. Spevak, July 1977

8. Computer Sciences Corporation, CSC/TM-81/6091, Software Engineering Laboratory (SEL) Programmer Workbench Phase I Evaluation, W. Decker, March 1981

9. Dolotta, T. A. and J. R. Mashay, Proceedings of the 2nd International Conference on Software Engineering, "An Introduction to the Programmer's Workbench," October 1976

10. Computer Sciences Corporation, (notes on a program under development), "Online Data Collection (ODC) Tool," C. E. Goorevich, 1980

11. Freeman, P., IEEE Computer Society, "Tutorial on Software Design Techniques: The Nature of Design," October 1976, pp. 35-63

12. Computer Sciences Corporation, CSC/TM-79/6263, Evaluation of the Caine, Faber, and Gordon Program Design Language (PDL) in the Goddard Space Flight Center (GSFC) Code 580 Software Development Environment, W. Decker, September 1979

REFERENCES (Cont'd)

13. Computer Sciences Corporation, CSC/TM-79/6012, FORTRAN Static Source Code Analyzer Design and Module Descriptions, E. M. O'Neill, January 1978

14. --, CSC/TM-77/6295, Manpower Allocation and Reporting System (MARS) Design Document, S. Waligora, October 1977

15. --, CSC/SD-78/6033, Financial Report Generation Program (FINREP) System Description and User's Guide, C. Rabbin, March 1978

16. Belady, L. A. and M. M. Lehman, "A Model for Large Program Development," IBM Systems Journal, 1976, vol. 15, no. 3, pp. 225-227

17. Digital Equipment Corporation, AA-D901A-TE, DECnet-VAX User's Guide, August 1978

BIBLIOGRAPHY OF SEL LITERATURE

Anderson, L., "SEL Library Software User's Guide," Computer Sciences-Technicolor Associates, Technical Memorandum, June 1980

Bailey, J. W., and V. R. Basili, "A Meta-Model for Software Development for Resource Expenditures," Proceedings of the Fifth International Conference on Software Engineering. New York: Computer Societies Press, 1981

Banks, F. K., "Configuration Analysis Tool (CAT) Design," Computer Sciences Corporation, Technical Memorandum, March 1980

Basili, V. R., "The Software Engineering Laboratory: Objectives," Proceedings of the Fifteenth Annual Conference on Computer Personnel Research, August 1977

Basili, V. R., "Models and Metrics for Software Management and Engineering," ASME Advances in Computer Technology, January 1980, vol. 1

Basili, V. R., "SEL Relationships for Programming Measurement and Estimation," University of Maryland, Technical Memorandum, October 1980

Basili, V. R., Tutorial on Models and Metrics for Software Management and Engineering. New York: Computer Societies Press, 1980 (also designated SEL-80-008)

Basili, V. R., and J. Beane, "Can the Parr Curve Help with the Manpower Distribution and Resource Estimation Problems?", Journal of Systems and Software, February 1981, vol. 2, no. 1

Basili, V. R., and K. Freburger, "Programming Measurement and Estimation in the Software Engineering Laboratory," Journal of Systems and Software, February 1981, vol. 2, no. 1

Basili, V. R., and T. Phillips, "Evaluating and Comparing Software Metrics in the Software Engineering Laboratory," Proceedings of the ACM SIGMETRICS Symposium/Workshop: Quality Metrics, March 1981

Basili, V. R., and T. Phillips, "Validating Metrics on Project Data," University of Maryland, Technical Memorandum, December 1981

Basili, V. R., and R. Reiter, "Evaluating Automatable Measures for Software Development," Proceedings of the Workshop on Quantitative Software Models for Reliability, Complexity and Cost, October 1979

Basili, V. R., and M. V. Zelkowitz, "Designing a Software Measurement Experiment," Proceedings of the Software Life Cycle Management Workshop, September 1977

Basili, V. R., and M. V. Zelkowitz, "Operation of the Software Engineering Laboratory," Proceedings of the Second Software Life Cycle Management Workshop, August 1978

Basili, V. R., and M. V. Zelkowitz, "Measuring Software Development Characteristics in the Local Environment," Computers and Structures, August 1978, vol. 10

Basili, V. R., and M. V. Zelkowitz, "Analyzing Medium Scale Software Development," Proceedings of the Third International Conference on Software Engineering. New York: Computer Societies Press, 1978

Chen, E., and M. V. Zelkowitz, "Use of Cluster Analysis To Evaluate Software Engineering Methodologies," Proceedings of the Fifth International Conference on Software Engineering. New York: Computer Societies Press, 1981

Church, V. E., "User's Guides for SEL PDP-11/70 Programs," Computer Sciences Corporation, Technical Memorandum, March 1980

Freburger, K., "A Model of the Software Life Cycle" (paper prepared for the University of Maryland, December 1978)

Higher Order Software, Inc., TR-9, A Demonstration of AXES for NAVPAK, M. Hamilton and S. Zeldin, September 1977 (also designated SEL-77-005)

Hislop, G., "Some Tests of Halstead Measures" (paper prepared for the University of Maryland, December 1978)

Lange, S. F., "A Child's Garden of Complexity Measures" (paper prepared for the University of Maryland, December 1978)

Miller, A. M., "A Survey of Several Reliability Models" (paper prepared for the University of Maryland, December 1978)

National Aeronautics and Space Administration (NASA), NASA Software Research Technology Workshop (proceedings), March 1980

Page, G., "Software Engineering Course Evaluation," Computer Sciences Corporation, Technical Memorandum, December 1977

Parr, F., and D. Weiss, "Concepts Used in the Change Report Form," NASA, Goddard Space Flight Center, Technical Memorandum, May 1978

Perricone, B. T., "Relationships Between Computer Software and Associated Errors: Empirical Investigation" (paper prepared for the University of Maryland, December 1981)

Reiter, R. W., "The Nature, Organization, Measurement, and Management of Software Complexity" (paper prepared for the University of Maryland, December 1976)

Scheffer, P. A., and C. E. Velez, "GSFC NAVPAK Design Higher Order Languages Study: Addendum," Martin Marietta Corporation, Technical Memorandum, September 1977

Software Engineering Laboratory, SEL-76-001, Proceedings From the First Summer Software Engineering Workshop, August 1976

--, SEL-77-001, The Software Engineering Laboratory, V. R. Basili, M. V. Zelkowitz, F. E. McGarry, et al., May 1977

--, SEL-77-002, Proceedings From the Second Summer Software Engineering Workshop, September 1977

--, SEL-77-003, Structured FORTRAN Preprocessor (SFORT), B. Chu, D. S. Wilson, and R. Beard, September 1977

--, SEL-77-004, GSFC NAVPAK Design Specifications Languages Study, P. A. Scheffer and C. E. Velez, October 1977

--, SEL-78-001, FORTRAN Static Source Code Analyzer (SAP) Design and Module Descriptions, E. M. O'Neill, S. R. Waligora, and C. E. Goorevich, January 1978

--, SEL-78-002, FORTRAN Static Source Code Analyzer (SAP) User's Guide, E. M. O'Neill, S. R. Waligora, and C. E. Goorevich, February 1978

--, SEL-78-003, Evaluation of Draper NAVPAK Software Design, K. Tasaki and F. E. McGarry, June 1978

--, SEL-78-004, Structured FORTRAN Preprocessor (SFORT) PDP-11/70 User's Guide, D. S. Wilson, B. Chu, and G. Page, September 1978

--, SEL-78-005, Proceedings From the Third Summer Software Engineering Workshop, September 1978

--, SEL-78-006, GSFC Software Engineering Research Requirements Analysis Study, P. A. Scheffer, November 1978

--, SEL-78-007, Applicability of the Rayleigh Curve to the SEL Environment, T. E. Mapp, December 1978

--, SEL-79-001, SIMPL-D Data Base Reference Manual, M. V. Zelkowitz, July 1979

--, SEL-79-002, The Software Engineering Laboratory: Relationship Equations, K. Freburger and V. R. Basili, May 1979

--, SEL-79-003, Common Software Module Repository (CSMR) System Description and User's Guide, C. E. Goorevich, S. R. Waligora, and A. L. Green, August 1979

--, SEL-79-004, Evaluation of the Caine, Farber, and Gordon Program Design Language (PDL) in the Goddard Space Flight Center (GSFC) Code 580 Software Design Environment, C. E. Goorevich, A. L. Green, and F. E. McGarry, September 1979

--, SEL-79-005, Proceedings From the Fourth Summer Software Engineering Workshop, November 1979

--, SEL-80-001, Configuration Analysis Tool (CAT) Functional Requirements/Specifications, F. K. Banks, C. E. Goorevich, and A. L. Green, February 1980

--, SEL-80-002, Multi-Level Expression Design Language-Requirement Level (MEDL-R) System Evaluation, W. J. Decker, C. E. Goorevich, and A. L. Green, May 1980

--, SEL-80-003, Multimission Modular Spacecraft Ground Support System (MMS/GSS) State-of-the-Art Computer System/Compatibility Study, T. Welden, M. McClellan, P. Liebertz, et al., May 1980

--, SEL-80-004, System Description and User's Guide for Code 580 Configuration Analysis Tool (CAT), F. K. Banks, W. J. Decker, J. G. Garrahan, et al., October 1980

--, SEL-80-005, A Study of the Musa Reliability Model, A. M. Miller, November 1980

--, SEL-80-006, Proceedings From the Fifth Annual Software Engineering Workshop, November 1980

--, SEL-80-007, An Appraisal of Selected Cost/Resource Estimation Models for Software Systems, J. F. Cook and F. E. McGarry, December 1980

--, SEL-81-001, Guide to Data Collection, V. E. Church, D. N. Card, F. E. McGarry, et al., September 1981

--, SEL-81-002, Software Engineering Laboratory (SEL) Data Base Organization and User's Guide, D. C. Wyckoff, G. Page, F. E. McGarry, et al., September 1981

--, SEL-81-003, Software Engineering Laboratory (SEL) Data Base Maintenance System (DBAM) User's Guide and System Description, D. N. Card, D. C. Wyckoff, G. Page, et al., September 1981

--, SEL-81-004, The Software Engineering Laboratory, D. N. Card, F. E. McGarry, G. Page, et al., September 1981

--, SEL-81-005, Standard Approach to Software Development, V. E. Church, F. E. McGarry, G. Page, et al., September 1981

--, SEL-81-006, Software Engineering Laboratory (SEL) Document Library (DOCLIB) System Description and User's Guide, W. Taylor and W. J. Decker, December 1981

--, SEL-81-007, Software Engineering Laboratory (SEL) Compendium of Tools, W. J. Decker, E. J. Smith, A. L. Green, et al., February 1981

--, SEL-81-008, Cost and Reliability Estimation Models (CAREM) User's Guide, J. F. Cook and E. Edwards, February 1981

--, SEL-81-009, Software Engineering Laboratory Programmer Workbench Phase 1 Evaluation, W. J. Decker, A. L. Green, and F. E. McGarry, March 1981

--, SEL-81-010, Performance and Evaluation of an Independent Software Verification and Integration Process, G. Page and F. E. McGarry, May 1981

--, SEL-81-011, Evaluating Software Development by Analysis of Change Data, D. M. Weiss, November 1981

--, SEL-81-012, Software Engineering Laboratory, G. O. Picasso, December 1981

--, SEL-81-013, Proceedings From the Sixth Annual Software Engineering Workshop, December 1981

--, SEL-81-014, Automated Collection of Software Engineering Data in the Software Engineering Laboratory (SEL), A. L. Green, W. J. Decker, and F. E. McGarry, September 1981

Turner, C., G. Caron, and G. Brement, "NASA/SEL Data Compendium," Data and Analysis Center for Software, Special Publication, April 1981

Turner, C., and G. Caron, "A Comparison of RADC and NASA/SEL Software Development Data," Data and Analysis Center for Software, Special Publication, May 1981

Weiss, D. M., "Error and Change Analysis," Naval Research Laboratory, Technical Memorandum, December 1977

Williamson, I. M., "Resource Model Testing and Information," Naval Research Laboratory, Technical Memorandum, July 1979

Zelkowitz, M. V., "Resource Estimation for Medium Scale Software Projects," Proceedings of the Twelfth Conference on the Interface of Statistics and Computer Science. New York: Computer Societies Press, 1979

Zelkowitz, M. V., and V. R. Basili, "Operational Aspects of a Software Measurement Facility," Proceedings of the Software Life Cycle Management Workshop, September 1977